Revision Control System for Large-scale Systems Management

Inventors

Robert Walsh and Bryan O'Sullivan

Background

Field of Invention

[0001]   The present invention relates generally to large-scale systems management, and more specifically to facilitating robust revision control of components of a large scale system.

Background of Invention

[0002]   Keeping track of multiple revisions to frequently changing computer configurations is a challenge facing system administrators and others responsible for maintaining large scale systems. Because of the number of files and other components involved in a large scale system, installing new software, upgrading existing software or modifying the configuration of the installed software can cause subtle changes in many of the configuration files which can go unnoticed. These changes to configuration files can then cause system problems that might not be noticed immediately. Thus, various nodes of a large scale system can end up with varying and possibly problematic configuration files which can be difficult to identify.

[0003]   Additionally, where multiple users make frequent updates to the configuration, it can be very difficult to ensure that each user has the latest version of

each system component. Users without current components are often unable to properly interact with the system. Furthermore, these users might subsequently make changes to old versions of components and propagate these changes to other users.

[0004] Changes to system configurations can involve much more than file modifications. Directories can be added or updated, hierarchies can be modified and file and directory ownership, group and permission information can be changed. Additionally, existing files and directories can be renamed or deleted.

[0005] What is needed are methods, systems and computer readable media to enable robust revision control of a system configuration, including directories, file and directory ownership, group and permission information. The methods, systems and computer readable media should also maintain revision history information across file and directory rename or delete operations.

Summary of Invention

[0006] Methods, systems and computer readable media facilitate robust revision control of components of a large scale system. A repository manager stores a plurality of system components under revision control in a master repository, according to internal names. The repository manager also stores binding information between internal names of system components and associated external names, as well as system component attribute information. The repository manager provides users with working areas, which can be synchronized with the master repository. Users are only able to access system components from working areas, according to their external names, via the repository manager. As users perform operations affecting system components, the repository

2

manager keeps track of the performed operations in a journal in the master repository. Responsive to a user committing a set of changes, the repository manager updates the master repository according to the performed operations as recorded in the journal, associates a new version number with the resulting state of the repository such that the change-set can be backed out, updates the journal to indicate that the operations have been committed to the repository, and synchronizes the user's working area with the updated repository.

[0007] The features and advantages described in this summary and the following detailed description are not all-inclusive, and particularly, many additional features and advantages will be apparent to one of ordinary skill in the art in view of the drawings, specification, and claims hereof. Moreover, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter, resort to the claims being necessary to determine such inventive subject matter.

Brief Description of the Drawings

[0008] Figure 1 is a block diagram illustrating a system 100 for practicing some embodiments of the present invention.

[0009] Figure 2 is a block diagram illustrating an example of a data record in a file storing both bindings and attributes concerning system components, according to one embodiment of the present invention.

[0010] Figure 3 is a block diagram illustrating an example of a section of a journal, according to one embodiment of the present invention.

[0011]   The figures depict embodiments of the present invention for purposes of

illustration only. One skilled in the art will readily recognize from the following

discussion that alternative embodiments of the structures and methods illustrated herein

may be employed without departing from the principles of the invention described herein.

Detailed Description of Preferred Embodiments

[0012]   Figure 1 illustrates a system 100 for practicing some embodiments of the

present invention. A repository manager 101 stores system components 103 (e.g.,

symbolic links, files, directories) in a revision control system 105 in a master repository

107, according to internal names. It is to be understood that although the repository

manager 101 is illustrated as a single entity, as the term is used herein a repository

manager 101 refers to a collection of functionalities which can be implemented as

software, hardware, firmware or any combination of the three. Where a repository

manager 101 is implemented as software, it can be implemented as a standalone program,

but can also be implemented in other ways, for example as part of a larger program, as a

plurality of separate programs, or as one or more statically or dynamically linked

libraries.

[0013]   An internal name of a system component 103 is a constant descriptor used

within the master repository 107 to denote the component 103, even as it is modified over

time. An external name is typically the name of the component 103 in the external file

system (for example, "/etc/ntp.conf"), and can change as the component 103 is modified.

For example, the file /etc/ntp.conf can be renamed, but the corresponding internal name

would not be changed. In one embodiment, the internal names used to describe the

system components 103 stored in the revision control system 105 are sequential numbers, such that the first system component 103 stored is assigned the internal name 0 (or some other number such as 1), and each subsequent system component 103 is assigned an incremented number. Of course, variations on the internal naming schemes are possible (e.g., the numbers can be decremented as additional system components 103 are added to the revision control system 105), all of which are within the scope of the present invention. Various internal naming formats will be apparent to those of ordinary skill in the relevant art in light of this specification.

[0014] In some embodiments, a next name indicator 109 is stored in the master repository 107. The next name indicator 109 can be, for example, a file containing the next available internal name (e.g., the next available sequential number). As system components 103 are added, the repository manager 101 reads the next name indicator 109, assigns the next available name to the system component 103 being added, and updates the next name indicator 109 accordingly.

[0015] It is to be understood that the stored system components 103 can comprise or describe the configuration of a large scale system, for example a computer cluster. The implementation mechanics of storing data in a revision control system 105 is known to those of ordinary skill in the relevant art, and thus the methodology for storing system components 103 will be apparent in light of this specification.

[0016] The repository manager 101 also stores, under revision control 105 in the master repository 107, a binding 111 between the internal name of each system component 103 and its external name, as well as attributes 113 concerning system components 103. An attribute 113 can be any relevant data concerning a component 103,

for example group or owner information concerning the component 103, the component 103 type, a version number, a mode, security settings or access permission data, or an indication that the component 103 has been deleted. Components 103 and their attributes 113 are discussed in greater detail below.

[0017]   Typically, the bindings 111 between internal and external names as well as the attributes concerning components 103 are stored in a single file under revision control 105 in the master repository 107. An example of such a file is illustrated in Figure 2, and described further below.

[0018]   As illustrated in Figure 1, users 115 can perform operations on system components 103. It is to be understood that a user 115 can be anyone accessing components 103 of the large scale system in question, for example administrators responsible for various parts of the system (e.g., individual nodes in a cluster system). The repository manager 101 provides a user 115 with a working area 117, which acts as a working copy of the master repository 107 with a copy of binding 111 and attribute 113 information, in which the user 115 can perform operations on system components 103, according to their external names. The repository manager 101 does not allow users 115 direct access to the master repository 107, but instead only allows users 115 to access system components 103 by their external names in a working area 117, via the repository manager 101. Specific operations that can be performed by users 115 and the relevance of performing such operations is discussed in greater detail below.

[0019]   The repository manager 101 also stores, in the master repository 107, a journal 119 of operations performed on system components 103 by users 115. When a user 115 performs an operation on a system component 103 according to its external

name, the repository manager 101 updates the journal 119 accordingly. For example, suppose a first user 115 operating in a first working area 117 installs a new software package, which creates various new directories and populates them with specific new files. The repository manager 101 will update the journal to indicate that the specific user 115 has added these system components 103. It is to be understood that the journal 119 resides in the master repository 107, is not copied to user 115 working areas 117, and is accessed directly only by the repository manager 101.

[0020]   Once the user 115 has completed making a set of changes (in the above example, installing the software package), the user commits the change-set to the master repository 107. To do so, the user 115 makes an appropriate indication to the repository manager 101, and in response the repository manager 101 reads the journal 119 and updates the revision control system 105 to reflect the set of changes made by the user 115. In the above example, the repository manager 101 would read the journal 119 and determine that the specific user 115 added a set of system components 103. Thus, the repository manager 101 would obtain an internal name for each new system component 103, and check it into the revision control system 105 under its internal name. The revision control system would update the bindings 111 to associate the internal names with their corresponding external names, and store appropriate attributes 113 concerning the new system components 103. The repository manager would also update the journal 119 to reflect that the user indication has been processed, and associate a new version number with the current state of the revision control system 105 (including the current state of the bindings 111 and attributes 113). In other words, the state of the revision control system is associated with a new version number after each set of user changes is

processed, such that the state of the associated system before and after the change set (in this example, before and after the software package is installed) can be recreated from the revision control system 105. Once a change-set has been processed, the repository manager 101 updates the user's 115 working area 117 to synchronize it with the master repository 107, providing an updated copy of the binding 111 and attribute 113 information.

[0021] As another example, suppose a second user 115, operating from a second working area 117, updates the software package previously installed by the first user, such that various existing system components 103 are modified. For example, the updating of the software package could result in various existing files being edited, renamed, deleted and moved to different locations. As these operations are performed by the second user 115 from the second working area 117, the repository manager 101 will update the journal 119 accordingly. Responsive to the indication from the second user 115 that the change set is complete, the repository manager 101 will read the journal 119 to determine which operations have been performed by the user 115 on which system components 103. The repository manager 101 reads the bindings 111 to determine the internal names of affected system components 103, updates the revision control system 105 to reflect the updates, and associates a new version number with the updated state of the revision control system 105.

[0022] It is to be understood that the specific format of the journal 119 can vary from embodiment to embodiment. One specific example is illustrated by Figure 3 and described in greater detail below. Other representation formats will be apparent to those of ordinary skill in the relevant art in light of this specification, and are within the scope

of the present invention. It is also to be understood that communication between a user

115 and the repository manager 101 can be via the user 115 executing a command on a

command line, via the 115 user operating a graphical user or other type of interface, or by

a process owned by the user 115 making an appropriate call to an application program

interface.

[0023]   Figure 2 illustrates an example of a data record 201 in a file storing both

bindings 111 and attributes 113 concerning system components 103, according to an

embodiment of the present invention. As explained above, bindings 111 and attributes

113 need not but can be stored in the same file. For purposes of visual clarity, Figure 2

illustrates an example of one possible format for storing information concerning one

component 103. Of course, typically bindings 111 and attributes 113 concerning many

components 103 would be stored.

[0024]   The illustrated binding 111 associated internal name "1" with the external

variable character string (in this case, a file name "a.txt" and a directory, 0). Note that the

external directory is stored as an internal name ("0"), indicating that the directory in

which the file a.txt is stored is itself a component 103 in the repository 107, with an

associated internal name. In some embodiments, the entire string "/0/a.txt" is stored in a

single field, called, for example, external name.

[0025]   As the corresponding system component 103 a.txt is modified, renamed,

moved and even deleted (and/or undeleted) by users 115, its external name will change,

but the internal name "1" will always be used to denote the component 103 in the revision

control system 105 in the master repository 107. Additionally, the directory in which

a.txt resides can be moved, renamed, etc. and its own internal name of 0 will remain

constant. When an external name has been changed by a user 115 performing an operation on a component 103, the repository manager 101 updates the appropriate stored binding 111 to reflect the change to the external name during the corresponding update of the revision control system 105 described above.

[0026] The stored attributes 113 illustrated in Figure 2 comprise a delete field, indicating that the component 103 has not been deleted, the component type 103 (in this example, a file), the user that currently owns the component 103, the relevant group, the current component 103 version number, and the access permissions. These attributes of course change as users 113 perform operations of the corresponding system component 103. When a user 115 performed operation has modified a component's attributes, the repository manager 101 updates the stored attributes 113 accordingly during the relevant revision control system 105 update. The attributes illustrated in Figure 2 are examples of possible attributes 113 concerning a system component 103. Of course, other examples are possible as desired.

[0027] Figure 3 illustrates an example of a section of a journal 119, according to an embodiment of the present invention. It is to be understood that the actual journal 119 need not be stored in human readable format. Figures 3 is intended simply to show types of data that can be stored in a journal, in conjunction with an explanation of their usages according to some embodiments of the present invention.

[0028] As illustrated in Figure 3, the journal contains entries pertaining to user 115 operations affecting system components 103. In the embodiment illustrated in Figure 3, the journal records, for each performed user 115 operation, the affected component 103, the specific performed operation, the name of the user 115 who performed the operation,

10

the associated working area 117, any arguments that the operation takes in addition to the component name (e.g., the new mode or owner if the operation is a change mode or change owner command), an optionally user 115 supplied comment associated with the operation and an indication of whether the operation has been committed. Committing operations is discussed in greater detail below. In some embodiments, the journal has fewer fields (e.g., no comment field, no commit field), more fields (e.g., a field describing order dependency in which multiple operations must be performed) or differently ordered or named fields, as desired.

[0029]   It is to be understood that users 115 can perform a wide variety of operations on system components 103 in order to add, modify and/or delete system configurations as desired. For example, a user 112 can add system components 103 to the master repository 107, edit, rename, delete and undelete components 103, modify linking information concerning components 103, change group, mode or owner information concerning components, undo earlier performed user 115 operations and back change-sets out of the master repository 107.

[0030]   Because user 115 operations are performed via the repository manager 101, the repository manger can update the journal 119 as the operations are performed. As described above, when a user has finished performing a set of operations (e.g., installing or modifying system components 103), the user 115 commits the change-set by sending an appropriate indication to the repository manager 107, which reads the journal 119 and updates the revision control system 105 according to that user's 115 uncommitted operations as documented in the journal 119. The repository manager 101 then updates the journal 119 to indicate that the operations have been committed, and synchronizes the

user's 115 working area 117 with the master repository 107. In some embodiments, the repository manager 101 updates the journal 119 to indicate that the operations have been committed by updating the appropriate commit fields in the journal 119. In other embodiments, the repository manager 101 updates the journal 119 to indicate that the operations have been committed by deleting the entries in the journal 119 relating to those operations.

[0031]   Of course, all operations that modify a system component 103 require that a user 115 have the system component 103 checked out of the master repository 107 for editing, and the repository manager 101 enforces this. Thus, a user 115 cannot edit, delete, rename or otherwise perform modifying operations on components 103 which are not checked out to the user 115. When a user 115 performs, for example, an edit operation, the repository manager 101 checks out the target component 103 out of the master repository 107 for the user 115. When the user 115 commits the edit, the repository manager 101 checks the edited version of the component into the revision control system 105, in the master repository 107. The repository manager 101 does not allow a user 115 to edit a component 103 checked out for editing by another user 115.

[0032]   Additionally, users 115 can request status information and the like from the repository manager 101. For example, a user 115 operating in a working area 117 can request and receive listings concerning change-sets entered into the revision control system 105 in the master repository 107, differences between two versions of a system component 103, information concerning system components 103, a listing of system components 103 currently checked out of the master repository 107 for editing and/or revision information concerning system components 103. Users 115 can also request and

receive information concerning their working areas 117, such as the location of a working area 117 (e.g., its absolute path in the relevant file system, or its path relative to the master repository 107) or the path to the root of the master repository 107. Additionally, a user 115 can also request the synchronization of an associated working area 117 with the master repository 107.

[0033]   As will be understood by those familiar with the art, the invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. Likewise, the particular naming and division of the modules, features, attributes, methodologies and other aspects are not mandatory or significant, and the mechanisms that implement the invention or its features may have different names, divisions and/or formats. Furthermore, as will be apparent to one of ordinary skill in the relevant art, the modules, features, attributes, methodologies and other aspects of the invention can be implemented as software, hardware, firmware or any combination of the three. Of course, wherever a component of the present invention is implemented as software, the component can be implemented as a standalone program, as part of a larger program, as a plurality of separate programs, as a statically or dynamically linked library, as a kernel loadable module, as a device driver, and/or in every and any other way known now or in the future to those of skill in the art of computer programming. Additionally, the present invention is in no way limited to implementation in any specific programming language, or for any specific operating system or environment. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.